Chair for Algebraic and Logical
Foundations of Computer Science
TU Dresden

Winter Term 2021/2022

Logic Programming Engineering – Assignment 3

Dr.-Ing. Sascha Klüppelholz

Exercise 3.1 [Primality test]

a) Write a Prolog predicate `prime/1` which is true for a given natural number n if the number is prime. If n is not prime, print out the smallest factor $k \in \mathbb{N}$, $k > 1$ that divides n, i.e., $n \bmod k \equiv 0$.

Examples:

```
?- prime(23).
true.
```

```
?- prime(143).
11 is a factor of 143.
false.
```

b) Provide a predicate `nthprime/2` that returns the n-th prime number. What is the 10001-th prime number?[1]

Example:

```
?- nthprime(6,X).
X = 13.
```

c) Provide a predicate `sumprimes/2` that allows computing the sum of all prime numbers below a given threshold. What is the sum of all prime numbers below two million?[2]

Example:

```
?- sumprimes(10,X).
X = 17.
```

[1] https://projecteuler.net/problem=7
[2] https://projecteuler.net/problem=10

Exercise 3.2 [Analyze list]

Create a new file `mylist.pl` and write a Prolog predicate `mylist_analyze/1` that takes a list as an argument and prints out the head and tail of the list together with the length of the list on the screen. If the given list is empty, the predicate should report on this fact. If the argument term is not a list the predicate should fail.

Examples:

```
?- mylist_analyze([a, b, c, d]).
This is the head of your list:  a.
This is the tail of your list:  [b, c, d].
The length of the list is 4.

?- mylist_analyze([]).
This is an empty list.
The length of the list is 0.

?- mylist_analyze(element).
No.
```

Exercise 3.3 [Integer ranges]

a) Provide a Prolog predicate `range/3` that generates all integers between a given lower and a given upper bound and puts them into a list. If the upper bound specified is less than the given lower bound, the empty list should be returned.

Examples:

```
?- range(3, 11, X).
X = [3, 4, 5, 6, 7, 8, 9, 10, 11].

?- range(7, 4, X).
X = [].
```

Provide two implementations of the above predicate: the first should be based on the lists predicate `append/3` and without using `between/3`, whereas the second should use `findall/3` or `bagof/3` predicate.

b) Now provide a predicate called `reverse_range/3` that creates a list in decreasing order rather than increasing order.

Example:

```
?- reverse_range(3, 11, X).
X = [11, 10, 9, 8, 7, 6, 5, 4, 3].
```

Again, provide two implementations, the first should be based on the lists predicate `append/3` similar to part a), whereas the second should make use of an accumulator, i.e., your predicate definition should start as follows:

```
reverse_range2(Start, End, RList) :-
    reverse_range(Start, End, [], RList).

reverse_range(Start, End, Accumulator, Accumulator) :-
    (Start>End), !.
```

```
reverse_range(Start, Start, [Start], [Start]) :- !.
```

c) Write a Prolog predicate `divisors/2` to compute the list of all divisors for a given natural number.

Example:

```
?- divisors(30, X).
X = [1, 2, 3, 5, 6, 10, 15, 30].
```

Again, provide two implementations of the above predicate: the first should be based on the lists predicate `append/3` and without using `between/3`, whereas the second should use `findall/3` or `bagof/3` predicate.